اولین همایش ملی فناوری اطلاعات و شبکه های کامپیوتری دانشگاه پیام نور
1th Payame Noor University National Conference on Information Technology and Networking
(PNUNCIT2013)
دانشگاه پیام نور – واحد طبس ۲۵ بهمن ماه ۱۳۹۱

بسمه تعالی

"پردازش داده های هواشناسی ایران بر اساس Cassandra"

نویسندگان: ۱- آنسه دانش آراسته، ۲- داود محمدپورزنجانی

(۱)دانشگاه آزاد اسلامی زنجان، دانشجوی کارشناسی ارشد گروه مهندسی کامپیوتر، زنجان، ج.ا.ایران ansehdanesh@yahoo.com

(۲)دانشگاه زنجان، عضو هیات علمی گروه علمی مهندسی کامپیوتر،، زنجان، ج.ا.ایران، dmp@znu.ac.ir

چکیده

Cassandra یک سیستم ذخیره سازی توزیع شده و Open Source برای مدیریت حجم عظیمی از داده‌ها در سر تا سر مراکز داده می-باشد. Cassandra یک ساختار ذخیره سازی Key-Value متوازن می‌باشد که در آن هر کلید به چندین مقدار نگاشت شده است. هدف Cassandra ارائه گسترش پذیری خطی و تحمل پذیری خطا بدون هیچ نقطه شکست واحد می‌باشد. Cassandra مدل داده BigTable Google به همراه زیرساختی شبیه Amazon Daynamo و سازگاری موزون را دارا می‌باشد. در مقاله حاضر قصد داریم نگاه جزیی تری به Cassandra داشته باشیم و پیاده سازی الگوریتم‌هایی از پردازش داده‌های حجیم هواشناسی در Cassandra با استفاده از زبان جاوا در مقایسه با RDBMSها بپردازیم.

واژه‌های کلیدی: Cassandra، BigTable، زبان برنامه نویسی Java، Amazon Daynamo.

۱-مقدمه

سیستم‌های ذخیره سازی NoSQL یک راه حل گسترش-پذیر و انعطاف پذیر را برای پایگاه داده‌های رابطه‌ای و سایر پایگاه داده‌ها ارائه می‌کند. Cassandra یکی از محبوب‌ترین این انتخاب‌هاست. Cassandra یک پیاده سازی از خانواده ستون گرای NoSQLهاست که مدل داده BigTable و معماری Amazon Daynamo را پشتیبانی می‌کند. برخی از خصوصیات Cassandra شامل [۱]:

- گسترش پذیری و دسترس پذیری بالا بدون هیچ نقطه شکست واحد.
- یک پیاده سازی از خانواده ستون‌های NoSQL.
- گذردهی بالای عمل write و گذردهی مناسب عمل read.
- سازگاری موزون و پشتیبانی از تکرار.
- شمای انعطاف پذیر.

این خصوصیات استفاده از Cassandra را به عنوان مدلی برای ذخیره و پردازش داده‌های حجیم آسان کرده است [۱].

Cassandra در بین نرم افزارهای مبتنی بر ذخیره ستونی جزء محبوب‌ترین‌هاست و کمی با آن‌ها متفاوت است. Cassandra داده را در چیزی شبیه به جدول Hash چند سطحی (چند بعدی) ذخیره می‌کند. در این نرم افزار، می-توانیم اطلاعات را بر مبنای کلیدها بازیابی کنیم و با آن مانند یک عنصر مقدار-کلید رفتار نماییم. از طرفی طبیعت چند بعدی Cassandra، استفاده از آن به عنوان "سوپر ستون‌ها" را برای ذخیره آیتم‌های چندگانه از نوع یکسان و ذخیره سازی مقدار-کلید آن‌ها در پایین‌ترین سطح را فراهم می‌نماید که انعطاف پذیری بالایی را فراهم می‌کند [۲].

Cassandra کلاینت‌های بسیاری در زبان‌های برنامه نویسی مختلف دارد. در این مقاله، قصد داریم پایگاه داده سازمان هواشناسی ایران را با استفاده از RDBMSها پیاده‌سازی کرده و سپس معادل آن‌را در NoSQL Cassandra و با استفاده از کلاینت جاوا پیاده سازی نماییم. برای این منظور باید مدل داده Cassandra که شامل Cluster، KeySpace، SuperColumnFamily، ColumnFamily، SuperColumn وColumn است را ایجاد کنیم.

برنامه‌ای که می‌نویسیم باید قادر به انجام اعمال زیر باشد [۳]:

۱- ایجاد ساختار پایگاه داده.
۲- وارد کردن داده‌ها در پایگاه داده.
۳- جست و جو برای وضعیت آب و هوای یک شهر داده شده.
۴- استخراج تحلیل‌های داده ای بر اساس پردازش‌های مورد نظر.

۲- تفاوت‌های طراحی در RDBMS و Cassandra

تفاوت‌های زیادی بین مدل Cassandra و روش‌های پرس و جوی آن در مقایسه با مدل RDBMS وجود دارد. این تفاوت‌ها عبارتند از [۳]:

عدم وجود زبان پرس و جو:

همان طور که می‌دانیم، SQL یک زبان پرس و جوی استاندارد برای پایگاه داده‌های رابطه‌ای می‌باشد. Cassandra هیچ زبان پرس و جوی خاصی ندارد و در عوض از طریق یک API که می‌توانیم به وسیله مکانیزم ترتیب RPC آن، یا thrift به آن دستیابی داشته باشیم، ارتباط برقرار می‌کند [۳].

عدم وجود جامعیت ارجاعی:

Cassandra مفهومی به نام جامعیت ارجاعی ندارد و از این رو، هیچ تصوری از Joinها ندارد. در یک پایگاه داده رابطه‌ای از کلید خارجی برای ارجاع دادن به کلید اصلی در یک رکورد در جداول مختلف استفاده می‌کنیم. اما، Cassandra این مفهوم را ندارد [۳].

شاخص‌های ثانویه:

Cassandra از شاخص‌های ثانویه استفاده نمی‌کند. در عوض، برای انجام این کار در Cassandra یک ColumnFamily ثانویه ایجاد کرده و داده‌های قابل جست و جو را در آن ذخیره می‌کنیم. در واقع، یک ColumnFamily برای ذخیره داده‌های مورد نظر ایجاد می‌کنیم و سپس آن‌ها را به IDهایشان نگاشت می‌دهیم. این ColumnFamily دوم به عنوان یک شاخص ثانوی صریح عمل می‌کند [۳].

مرتب‌سازی، الگوی طراحی:

در RDBMSها به راحتی می‌توانیم با استفاده از دستور ORDER BY در query ترتیب نمایش رکوردها را تغییر دهیم. در Cassandra هیچ پشتیبانی برای دستورات Order By و Group By وجود ندارد. در Cassandra مرتب سازی به گونه‌ای متفاوت عمل می‌کند. در این پایگاه داده، تعاریف ColumnFamilyها شامل یک عنصر Compare With است که ترتیبی را که در آن سطرها بر روی readها مرتب می‌شوند را نشان می‌دهد. اما برای هر query پیکربندی نشده است. به هر حال، دستوری به نام SliceRange وجود دارد که تقریباً مشابه Order By در SQL عمل می‌کند [۳].

Denormalization:

در طراحی پایگاه داده‌های رابطه‌ای به اهمیت نرمال سازی توجه می‌کنیم. اما هنگام کار با Cassnadra این کار ایده خوبی نخواهد بود، چون Cassandra زمانی که مدل داده غیر نرمال باشد بهینه‌تر کار می‌کند [۳].

در Cassandra غیرنرمال‌سازی کاملاً طبیعی است، چون مدل داده ساده است. نکته مهم این است که در Cassandra به جای مدلسازی داده و سپس نوشتن query، ابتدا queryها را مدلسازی کرده و سپس داده‌ها را در آن‌ها سازماندهی می‌کنیم [۳].

روش‌های مختلفی برای انتقال داده از ساختار داده رابطه‌ای به ساختار Cassandra وجود دارد. اما این انتقالات، عموماً شامل دستورات پیچیده است. در حالت استفاده از داده ساخته شده در بار گزارنده‌ها داده پردازش شده می‌تواند به فرمت JSON استخراج شده و سپس به ساختار داده Cassandra با استفاده از این بار گزارنده‌ها، بار گزاری شود [۴].

بار گزارنده‌های سفارشی می‌توانند در حالت توزیع‌های اضافی ایجاد شوند، که می‌تواند داده را در انبار پردازش شده و یا فایل‌های JSON تعامل دهند. بنابراین، رویکرد کلی انتقال شامل مراحل زیر است [۴]:

- آماده سازی داده، به ازای هر فرمت فایل JSON [۴].

- استخراج داده به فایل‌های flate به ازای هر فرمت فایل JSON یا استخراج داده از انبار داده پردازش شده با استفاده از بار گزارنده‌های داده سفارشی [۴].
- بار گزاری داده با استفاده از بار گزارنده‌های داخلی یا سفارشی در ساختار داده Cassandra [۴].

۳- پایگاه داده هواشناسی

در این مقاله، قصد داریم مثال ساده‌ای از پایگاه داده هواشناسی ایران ارائه دهیم و نحوه ذخیره سازی داده در مدل رابطه‌ای و به دنبال آن معادل مدل داده Cassandra آن را نمایش دهیم. همچنین با استفاده از کدهای Java داده‌هایی را به پایگاه داده اضافه کرده و همچنین یک بررسی بر کلاینت Java خواهیم داشت [۵].

۳-۱- مدل‌سازی با استفاده از RDBMS و Cassansra

برای شروع کار یک نمایش جدولی از پایگاه داده مورد نظر با استفاده از RDBMSها ایجاد می‌کنیم و سپس معادل آن جدول را در Cassandra نیز ایجاد می‌کنیم. برای مدلسازی پایگاه داده مورد نظر با RDBMSها ابتدا یک ساختار جدولی از پایگاه داده ایجاد می‌کنیم. جداول ۱ تا ۵ ساختار RDBMS پایگاه داده هواشناسی را نشان می‌دهد.

نام جدول: جدول استان‌ها		
نام فیلد	نوع داده ای	ویژگی
کد استان	عددی	کلید اصلی
نام استان	رشته ای	

جدول ۱ مدل RDBMS جدول استان‌های پایگاه داده هواشناسی ایران

نام جدول: جدول شهرستان‌ها		
نام فیلد	نوع داده ای	ویژگی
کد شهرستان	عددی	کلید اصلی
کد استان	عددی	کلید خارجی
نام شهرستان	رشته ای	
جمعیت	عددی	

جدول ۲ مدل RDBMS جدول شهرستان‌های پایگاه داده هواشناسی ایران

نام جدول: جدول ایستگاه های هواشناسی		
نام فیلد	نوع داده ای	ویژگی
کد ایستگاه	عددی	کلید اصلی
کد شهرستان	عددی	کلید خارجی

نام ایستگاه	رشته ای	
محل ایستگاه	رشته ای	
مسئول ایستگاه	رشته ای	

جدول ۳ مدل RDBMS جدول ایستگاه‌های هواشناسی

کد ایستگاه	عددی	کلید خارجی
تاریخ برداشت	تاریخ	
میزان بارندگی	عددی	
ساعات آفتابی	عددی	
ساعات بارندگی	عددی	
ساعات ابری	عددی	

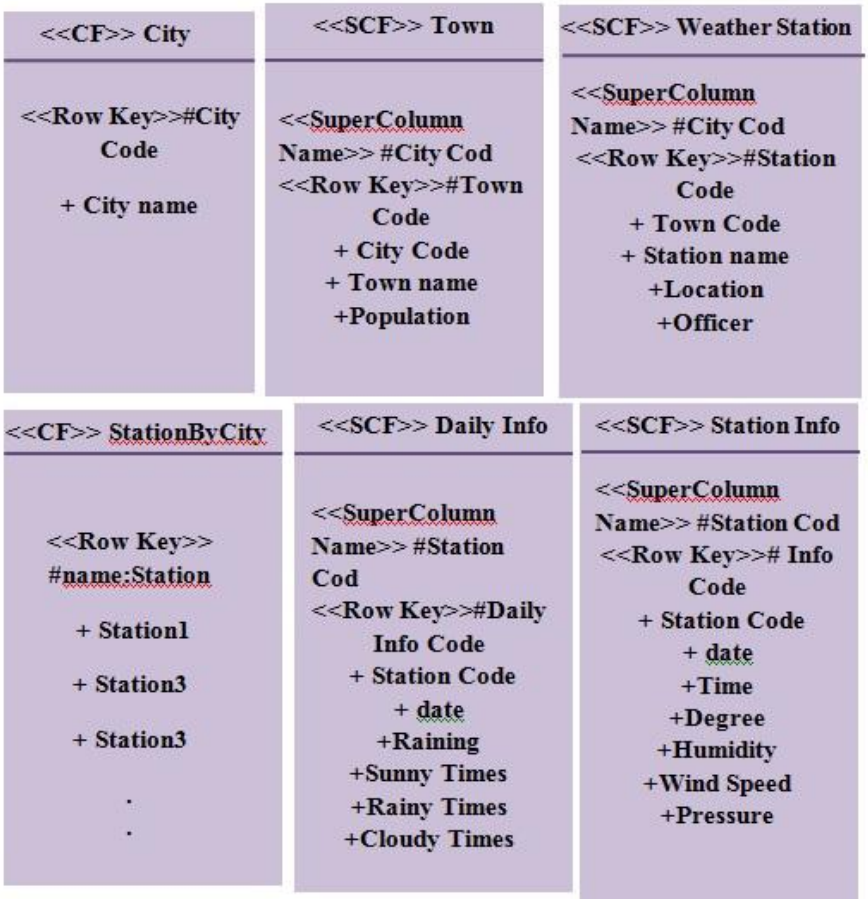
جدول ۵ مدل RDBMS جدول برداشت‌های روزانه پایگاه داده هواشناسی

نام جدول: جدول اطلاعات برداشت‌های لحظه ای ایستگاه های هواشناسی		
نام فیلد	نوع داده ای	ویژگی
کد برداشت	عددی	کلید اصلی
کد ایستگاه	عددی	کلید خارجی
تاریخ برداشت	تاریخ	
زمان برداشت	زمان	
درجه دما	عددی	
رطوبت	عددی	
سرعت باد	عددی	
فشار	عددی	

جدول ۴ مدل RDBMS جدول اطلاعات لحظه‌ای پایگاه داده

مدل رابطه‌ای ما شامل مجموعه‌ای از جدول‌ها می‌باشند که بر اساس تعریف کلیدهای خارجی به هم مربوط شده‌اند. معادل این جدول در مدل پایگاه دادهٔ NoSQL Cassandra به صورت شکل ۲ می‌باشد [۳].

نام جدول: جدول اطلاعات برداشت‌های روزانه ایستگاه های هواشناسی		
نام فیلد	نوع داده ای	ویژگی
کد برداشت روزانه	عددی	کلید اصلی



شکل ۲ معادل Cassandra مدل RDBMS پایگاه داده هواشناسی ایران

در مدل RDBMS می‌توانیم با استفاده از پرس و جوهای SQL ایستگاهی را در یک شهر جست و جو کنیم. اما در Cassandra به دلیل اینکه ساختار SQLی نداریم، شاخصی به نام *StatinoByCity* به شکل ColumnFamily برای این منظور، ایجاد می‌کنیم [۳].

۳-۲- ایجاد ساختار ذخیره سازی در Cassandra

برای پیاده سازی ساختار ارائه شده در بخش قبل در Cassandra با توجه به شکل ۲، در ابتدا نیاز به ایجاد مدل داده داریم. برای این منظور باید مدل داده Cassandra را که شامل کلاستر، KeySpace، SperColumnFamily، ColumnFamily، SuperColumn Column است را ایجاد

کنیم. برای این کار ابتدا از ساختن KeySpace شروع می‌کنیم که ColumnFamilyها و سایر تنظیمات را توصیف می‌کند. این کار باید درون هر گره در خوشه انجام شود [۶].

```
keyspaces:

- name: Weather
replica_placement_strategy:
org.apache.cassandra.locator.RackUnawareStratey
replication_factor: ۱
column_families:
  - name: City
  compare_with: UTF۸Type
  name:Station/City
  compare_with: UTF۸Type
  - name: Town
  column_type: Super
  compare_with: UTF۸Type
  compare_subcolumns_with: UTF۸Type
  - name: WeatherStation
  column_type: Super
  compare_with: BytesType
  compare_subcolumns_with: BytesType
  - name: StationByCity
  compare_with: UTF۸Type
- name: DailyInfo
  column_type: Super
  compare_with: UTF۸Type
  compare_subcolumns_with: UTF۸Type
- name: StationInfo
  column_type: Super
  compare_with: UTF۸Type
  compare_subcolumns_with: UTF۸Type
```

برنامه ما نیازمند یک ساختار داده استاندارد است که به عنوان یک انتقال دهند اشیاء عمل کند. چنین ساختارهایی برای سازماندهی کردن اشیاء لازم است [۶]. در اینجا ساختار داده هواشناسی را برای نگهداری اطلاعات مربوطه ایجاد می‌کنیم.

```
package com.cassandraguide.Weather;
public class Weather {
public String id;
public String city;
}
```

۳-۳- اتصال به Cassandra از زبان برنامه نویسی Java

پس از ایجاد ساختار داده اولیه و تعاریف KeySpaceها و ColumnFamilyها، نوبت به اتصال به این ساختار داده‌ها از زبان برنامه‌نویسی است. در اینجا ما زبان برنامه نویسی Java را برای این کار انتخاب کردیم. اتصال به زبان برنامه نویسی جاوا توسط کد زیر صورت می‌گیرد [۳]:

```
public class Connector {
TTransport tr = new TSocket("localhost", ۹۱۶۰);
public Cassandra.Client connect() throws
TTransportException,
TException, InvalidRequestException {
TFramedTransport tf = new TFramedTransport(tr);
TProtocol proto = new TBinaryProtocol(tf);
Cassandra.Client client = new
Cassandra.Client(proto);
tr.open();
client.set_keyspace(KEYSPACE);
return client;
}
public void close() {
tr.close();
}
}
```

حال نوبت به وارد کردن داده در پایگاه داده می‌رسد. البته قبل از آن بهتر است توسط کد زیر توابع جاوا که مورد نیاز است را فراخوانی کنیم.

```
package com.cassandraguide.hotel;
import static
com.cassandraguide.hotel.Constants.CAMBRIA_NA
ME;
```

```
import static
com.cassandraguide.hotel.Constants.CL;
import static
com.cassandraguide.hotel.Constants.CLARION_NA
ME;
import static
com.cassandraguide.hotel.Constants.UTF۸;
import static
com.cassandraguide.hotel.Constants.WALDORF_N
AME;
import static
com.cassandraguide.hotel.Constants.W_NAME;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.apache.cassandra.thrift.Cassandra;
import org.apache.cassandra.thrift.Clock;
import org.apache.cassandra.thrift.Column;
import
org.apache.cassandra.thrift.ColumnOrSuperColumn;
import org.apache.cassandra.thrift.ColumnParent;
import org.apache.cassandra.thrift.ColumnPath;
import org.apache.cassandra.thrift.Mutation;
import org.apache.cassandra.thrift.SuperColumn;
import org.apache.log۴j.Logger;
```

بعد از آن بایـد KeySpaceای را کـه ایجـاد کـردیم تنظـیم کنیم که برای این کار از تکه کد زیر استفاده می‌کنیم:

```
Keyspaceks =client.getKeyspace("Weather");
```

حال فرض کنید که می‌خواهیم رکورد جدیدی را، مثلاً یـک شهر جدید، به پایگاه داده خود اضافه کنیم. برای این منظور در ابتدا یک Column Path جدید به سـتون "شـهر" اضـافه می‌کنیم.

```
ColumnPath columnPath= new ColumnPath
("Town") ;
```

حالا تنها چیزی که می‌خواهیم، اضافه کردن یک مقـدار بـه ستون است که این مقادیر باید به صورت جفـت‌هـای Key-Value اضافه شوند. بـرای مثـال فـرض کنیـد مـی‌خـواهیم مقادیر اضافه شده به صورت زیر باشند [۶]:

City num: ۵
Town num: ۱
Town name: زنجان
Population: ۱۰۰۰

کلید در اینجا کد شهرستان اسـت و کـد اسـتان: Data، نـام شهرستان: Data و جمعیت: Data ستون‌های آن مـی‌باشـند. بنابراین، برای اضافه کردن کلید "کد شهرستان" به صـورت زیر عمل می‌کنیم [۶]:

```
String Rowkey="Town num";
String SuperCFName="Town name";
String value = "Zanjan";
columnPath.setColumn(columnName.getBytes());
ks.insert(key, columnPath, value.getBytes());
```

تا اینجا ستون نام شهرستان را با داده‌ها پر کرده و بـه کلیـد شماره شهرستان اضـافه کـردیم. همـین کـار را بـرای سـایر ستون‌ها نیز انجام می‌دهیم.

```
String columnName="Pop";
String value = "۱۰۰۰";
columnPath.setColumn(columnName.getBytes());
ks.insert(key, columnPath, value.getBytes());
```

برای اضافه کردن رکورد جدید، تنها کافیست کلید را عـوض کنیم و فیلدهای مربوطه را اضافه کنیم. برای جست و جوی مشخصات آب و هوایی یک شهر از طریق جست و جـوی آن شهر نیز از کد زیر استفاده می‌کنیم [۳].

```
List<City> City = app.WeatherByCity("Zanjan",
"ZJ");
City C = City.get(۰);
LOG.debug("You picked " + C.name);
LOG.debug("Found Weather in city. Results: " +
Weather.size());
LOG.debug("All done.");
```

۴- MapReduce

MapReduce متدی برای استاندارد سازی روش‌های پیاده سازی داده‌های موازی انبوه در پردازش‌های شبکه‌ای می‌باشد. داده‌های ورودی به چندین بلوک پردازشی شکسته می‌شوند که هر بلوک توسط یک تابع map() پردازش می-شود و سپس نتایج تمامی بلوک‌ها توسط تابع reduce() ترکیب می‌شوند [۷].

برای محاسبات آماری مجموعه داده‌های بزرگ، مانند فرکانس، میانگین و ... یک پیاده سازی مبتنی بر MapReduce می‌تواند داده‌ها را قطعه بندی کرده و قطعات را به صورت موازی اجرا کند. حجیم شدن پردازش-های موازی معمولاً در مرحله map رخ می‌دهد. مرحله reduce معمولاً مراحل را برای ترکیب نتایج محاسباتی مستقل، کاهش می‌دهد [۷].

۵- پیاده سازی پردازش‌های داده‌ای

پس از پر کردن پایگاه داده با مقادیر و ایجاد ColumnFamilyها و SuperColumFamillyها نوبت به پیاده سازی پردازش‌های داده‌ای مورد نیاز می‌رسد. برای پیاده سازی برنامه‌ها از زبان Java استفاده شده است و داده-ها به صورت فرضی می‌باشند.

۵-۱- پیاده سازی ساختار عمومی برنامه

برای بخش اصلی برنامه و ایجاد بدنه عمومی برنامه کد زیر را استفاده می‌کنیم [۳]. پیاده سازی ساختار اصلی برنامه با استفاده از تابع MapReduce انجام می‌گیرد. تابع map() عملیات میانگین، سیگما و ... را محاسبه می‌کند و تابع reduce() این مقادیر را به صورت موازی برای دستیابی به فرمول اصلی، ترکیب می‌کند.

```
public class WeatherApp {
private static final Logger LOG =
Logger.getLogger(WeatherApp.class);
 public static void main(String[] args) throws
Exception {
new Prepopulate().prepopulate();
LOG.debug("** Database filled. **");
LOG.debug("** Starting Weather app. **");
WeatherApp app = new WeatherApp();

SlicePredicate predicate = new SlicePredicate();
SliceRange sliceRange = new SliceRange();
sliceRange.setStart(Weather.getBytes());
sliceRange.setFinish(Weather.getBytes());
predicate.setSlice_range(sliceRange);
String scFamily = "Town";
ColumnParent parent = new
ColumnParent(scFamily);
KeyRange keyRange = new KeyRange();
keyRange.start_key = "".getBytes();
keyRange.end_key = "".getBytes();
List<Town> Towns = new ArrayList<Town>();
Connector cl = new Connector();
Cassandra.Client client = cl.connect();
List<KeySlice> slices = client.get_range_slices(
parent, predicate, keyRange, CL);
for (KeySlice slice : slices) {
List<ColumnOrSuperColumn> cols = slice.columns;
TOWN T = new Town();
poi.name = new String(slice.key);
for (ColumnOrSuperColumn cosc : cols) {
SuperColumn sc = cosc.super_column;
List<Column> colsInSc = sc.columns;
for (Column c : colsInSc) {
String colName = new String(c.name, UTF^);
if (colName.equals("desc")) {
T.desc = new String(c.value, UTF^);
}
if (colName.equals("Town")) {
T.Town = new String(c.value, UTF^);
}
}
LOG.debug("Found: " + T.name +
". \nnum: " + T.Population +
". \nPop: " + T.phone);
```

ساختار کلی معادله فوق را به زبان جاوا و با استفاده از پیاده سازی MapReduce در Cassandra پیاده سازی می‌کنیم.

```
public class CClient {
   public static final String UTF۸ = "UTF۸";

   public static void main(String[] args) throws
UnsupportedEncodingException,
         InvalidRequestException,
UnavailableException, TimedOutException,
         NotFoundException,
CharacterCodingException, TException {

      String keyspace = "CassandraTest";
      String columnFamily = "MyColumnFamily";

      TFramedTransport transport = new
TFramedTransport(new TSocket(
            "localhost", ۹۱۶۰));
      Cassandra.Client client = new
Cassandra.Client(new TBinaryProtocol(
            transport));
      transport.open();

      client.set_keyspace(keyspace);

      String keyUserID = "۱۱";

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

SlicePredicate predicate = new SlicePredicate();
      SliceRange sliceRange = new SliceRange();
      sliceRange.setStart(new byte[۰]);
      sliceRange.setFinish(new byte[۰]);
      predicate.setSlice_range(sliceRange);

      System.out.println("\nrow:");

      List<ColumnOrSuperColumn> results = null;
      results =
client.get_slice(ByteBuffer.wrap(keyUserID.getByte
s(UTF۸)), columnParent, predicate,
ConsistencyLevel.ONE);
      for (ColumnOrSuperColumn result : results) {
         column = result.column;
         System.out.println(new
String(column.getName()) + " -> "
T.add(T);
}
}
cl.close();
return T;
}
```

۵-۲- پیاده سازی الگوریتم تجزیه و تحلیل آماری از دما

همان طور که سازمان جهانی هواشناسی (۱۹۹۶) بیان کرد، روند داده‌های اقلیمی به ندرت خطی است. از این رو، سازمان مزبور به جای به‌کارگیری روش‌های خطی برآورد روند، روش‌های غیرخطی و رتبه‌ای را پیشنهاد می‌کند [۸].

بر این اساس، در این مقاله برای شناخت چرخه‌های موجود در مقادیر دما از خود همبستگی استفاده می‌کنیم تا نوسانات دوره‌ای و چرخه‌های موجود مورد ارزیابی قرار گیرند. تابع r(k) رابط خطی موجود بین مشاهدات سری زمانی را که با k وقفه زمانی جدا شده‌اند را اندازه گیری می‌کند [۸].

$$r(k)=\frac{\sum_{i=0}^{n+1}(x_i-\bar{x})(x_i+1-\bar{x})}{\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2\sum(x_i+1-\bar{x})^2}}$$

مقدار r(k) بین ۱+ و ۱- می‌باشد. آن دسته از مقادیر r(k) که به ۱+ نزدیک‌ترند، نشان دهند این است که مشاهداتی که با k وقفه زمانی جدا شده‌اند، به حرکت با یکدیگر در مسیری خطی با شیب مثبت داشته و نوسانات k ساله را می‌توان از آن تعبیر نمود [۸].

معادله فوق یک معادله پایه‌ای برای محاسبه مقادیر دما و نوسانات دوره‌ای آن به حساب می‌آید. بنابراین، در این مقاله

```
public class LocalMapReduce<TMapInput,
TMapOutput, TOutput> {
   private int m_threads;
   private Mapper<TMapInput, TMapOutput>
m_mapper;
   private Reducer<TMapOutput, TOutput>
m_reducer;
   ...
   public TOutput mapReduce(Iterator<TMapInput>
inputIterator) {
      ExecutorService pool =
Executors.newFixedThreadPool(m_threads);
      Set<Future<TMapOutput>> futureSet = new
HashSet<Future<TMapOutput>>();
      while (inputIterator.hasNext()) {
         TMapInput m = inputIterator.next();
         Future<TMapOutput> f =
pool.submit(m_mapper.makeWorker(m));
         futureSet.add(f);
         Thread.sleep(۱۰);
      }
      while (!futureSet.isEmpty()) {
         Thread.sleep(۵);
         for (Iterator<Future<TMapOutput>> fit =
futureSet.iterator(); fit.hasNext();) {
            Future<TMapOutput> f = fit.next();
            if (f.isDone()) {
               fit.remove();
               TMapOutput x = f.get();
               m_reducer.reduce(x);
            }
         }
      }
      return m_reducer.getResult();
   }
}
```

```
      + new String(column.getValue()));
      }
      transport.close();
   }
import Temp
def mapFUNC(row):
    return (Temp.mean(row));
def reduceFUNC(row۱, row۲):
   ////محاسبه میانگین برای دو سطر از داده
  n_a, mean_a= row۱
  n_b, mean_b = row۲
  n_ab = n_a + n_b
  mean_ab = ((mean_a * n_a) + (mean_b * n_b)) /
n_ab
return (n_ab, mean_ab, var_ab)
numRows = ۱۰۰;
numSamplesPerRow = ۵۰۰
x = Temp.random.rand(numRows,
numSamplesPerRow
y = reduce(reduceFunc, map(mapFunc, x))
print "n=%d, mean=%f" % y

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
{
val SelfColleration = …
//صورت کسر
val Sigma- numerator = get_sigma((x –mean) * (
x+۱-mean));
//مخرج کسر
 val Sigma- denominator = get_sigma(math.sqrt(
pow((x –mean),۲) *(pow(( x+۱-mean),۲)));
r(k)= Sigma- numerator / Sigma-denominator
}
```

پس از محاسبه فرمول کلی با استفاده از توابع map و reduce، حال باید این الگوی MapReducی را با استفاده از زبان جاوا و Cassandra پیاده سازی نماییم.

//پیاده سازی الگوی mapreducی در cassandra

۶- نتیجه گیری

در این مقاله چگونگی ایجاد یک پایگاه داده کامل در Cassandra را نشان دادیم. مدل داده جدولی پایگاه داده در RDBMS و معادل آن در Cassandra را نشان دادیم و نحوه اضافه کردن ColumnFamilyها، SuoercolumnFamilyها و همچنین اضافه کردن مقدار به این فیلدها در Cassandra

را بررسی کردیم. برای اجرای پایگاه داده ایجاد شده در زبان جاوا با استفاده از thrift، که یک فرایند اتصال چند مرحله‌ای است ابتدا به پایگاه داده Cassandra متصل شدیم و سپس با استفاده از کدهای جاوا مقادیر را درون ساختار ایجاد شده در Cassandra وارد کردیم.

در نهایت با استفاده از فرمول پایه‌ای همبستگی برای محاسبه دما و نوسانات دوره‌ای آن، مدل MapReduce آن را طراحی کرده و با استفاده از Cassandra و زبان جاوا پیاده سازی نمودیم.

مراجع

[۱] Perera S, Consider the Apache Cassandra database, Jul ۲۰۱۲.

[۲] Reuven M. Lerner, At the Forge, *Cassandra [Internet]*, Volume ۲۰۱۰ Issue ۱۹۸, October ۲۰۱۰, http://*dl.acm.org/citation.cfm?id= ۱۸۸۳۵۴۴.*

[۳] Hewitt E, Cassandra; The Definitive Guide, United States of America, O'Reilly Media, Inc., ۲۰۱۰.

[۴] Phani Krishna Kollapur Gandla, Migration of Relational Data structure to Cassandra (No SQL) Data structure, Nov ۲۰۱۱, http://www.codeproject.com/Articles/۲۷۹۹۴۷/Migration-of-Relational-Data-structure-to-Cassandr.

[۵] Salmon Run, Modeling Relational data With Cassandra, October ۲۰۱۰, http://sujitpal.blogspot.com/۲۰۱۰/۱۰/modeling-relational-data-with-cassandra.html.

[۶] Reuven M. Lerner, Starting to write a Cassandra app in Java, July ۲۰۱۰,http://ac۳۱۰۰۴.blogspot.com/۲۰۱۰/۷/starting-to-write-cassandra-app-in-java.html.

[۷] Seibel P, Calculating variance and mean with MapReduce], Jun ۲۰۱۰, http://blog.cordiner.net/۲۰۱۰/۶/۱۶/calculating-variance-and-mean-with-mapreduce-python/l.

[۸] اکبرزاده یونس، حسین بابئی مصطفی، صادقی شقاقی فاطمه، تحلیل آماری بر میانگین سالانهٔ دمای شهر تبریز طی دورهٔ آماری (۲۰۰۷ – ۱۹۵۱)،
http://www.eaz.ir/farsi/index.php?option=com_content&view=article&id=۲۷۱۹۴&Itemid=